



FIWARE ZONE

Práctica IoT

<https://www.fiware.org>

<https://lab.fiware.org>

Follow @FIWARE on Twitter!

<https://fiware.zone>

Follow @FIWAREZone on Twitter!

Telefónica



JUNTA DE ANDALUCÍA
CONSEJERÍA DE EMPLEO, EMPRESA Y COMERCIO

Conocimientos previos

Se emplearán las siguientes tecnologías, por lo que es conveniente tener un conocimiento previo de:

- **C++:** Para la programación del microcontrolador (Arduino)
- **HTML:** Para la página web y los formularios que crea el programa
- **API REST:** Para gestionar la API de FIWARE
- **FIWARE – IoT Agent Ultraligh 2.0:** Para el envío de datos por parte del dispositivo

01



Descripción del Hardware

Placa de desarrollo

Specs

ESP8266

RAM: 128KB

ROM: 4MB

Alimentación: 3.3V

WiFi integrado

13 Pin IO

USB<->UART incluido

Regulador 5V (USB) a 3.3V



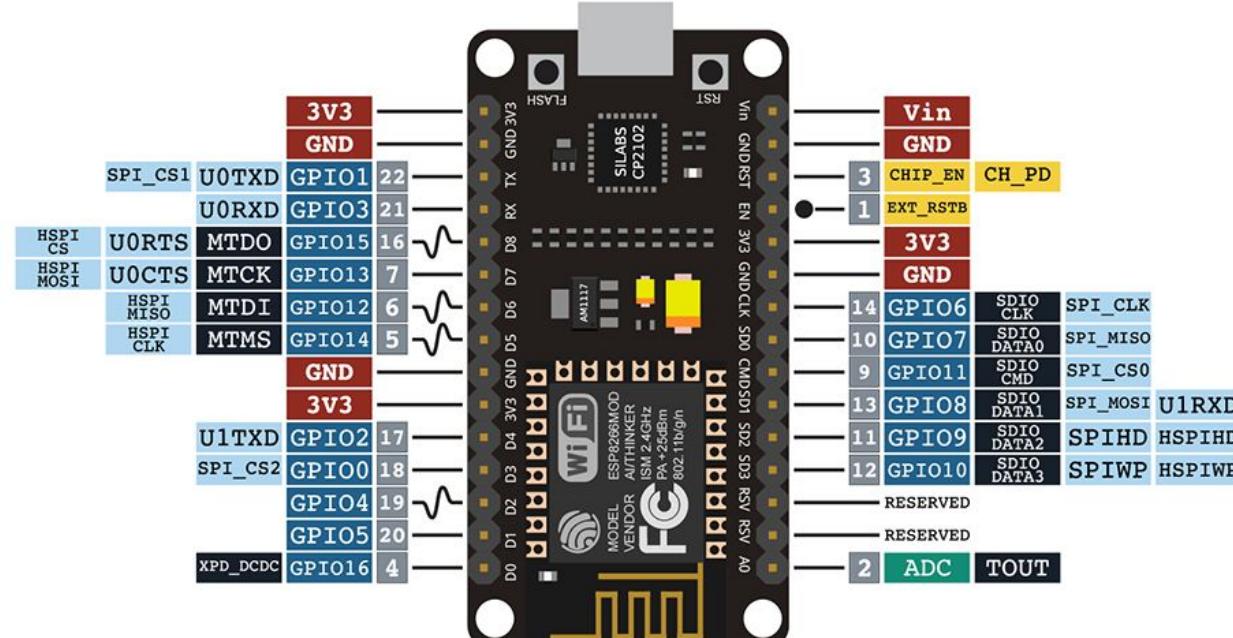
Pinout

NOTES:

- ⚠ Typ. pin current 6mA (Max. 12mA)
- ⚠ For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
- ⚠ On boot/reset/wakeup, keep GPIO15 LOW and GPIO2 HIGH.



FIWARE
 ZONE



POWER	SP. FUNCTION(S)
I/O	COMM. INTERFACE
ADC	PIN NUMBER
CONTROL	PWM
N/C	

Telefonica

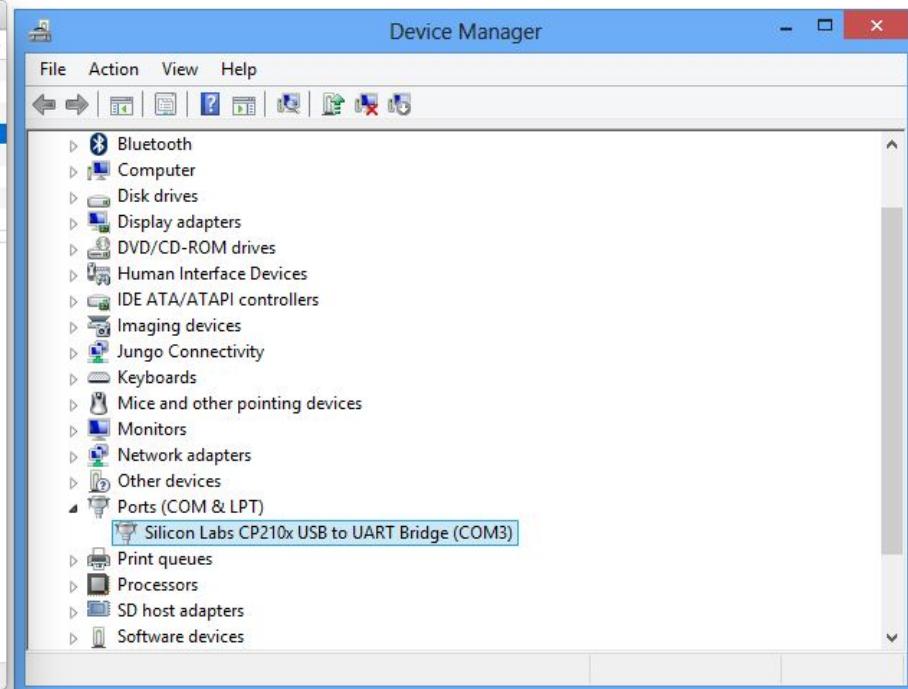
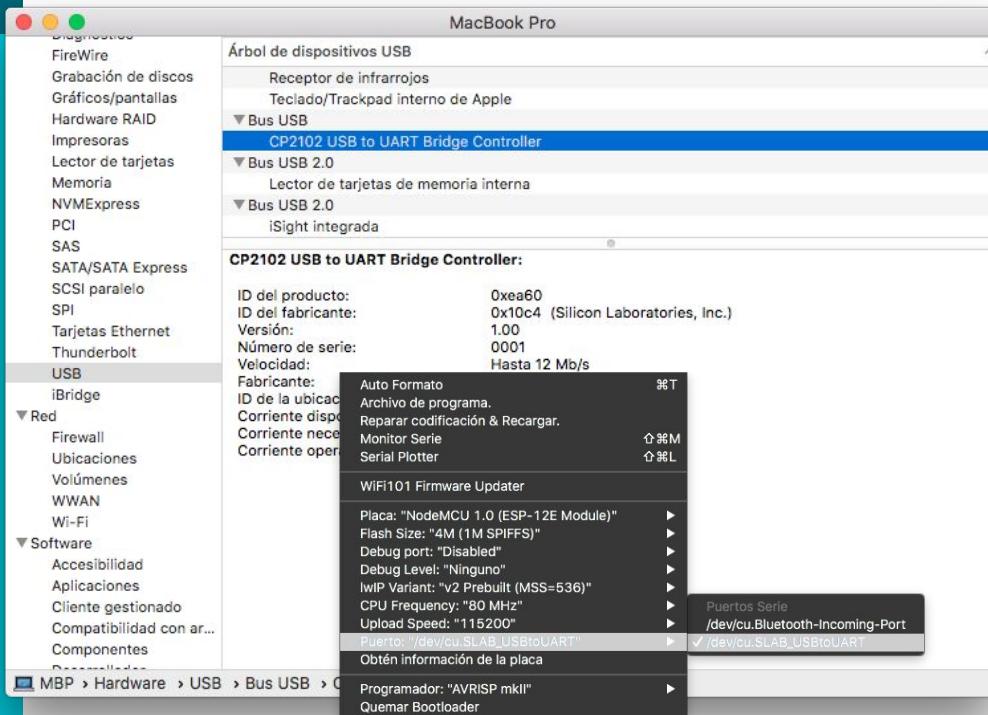


02



Instalación del IDE

2.1 Drivers

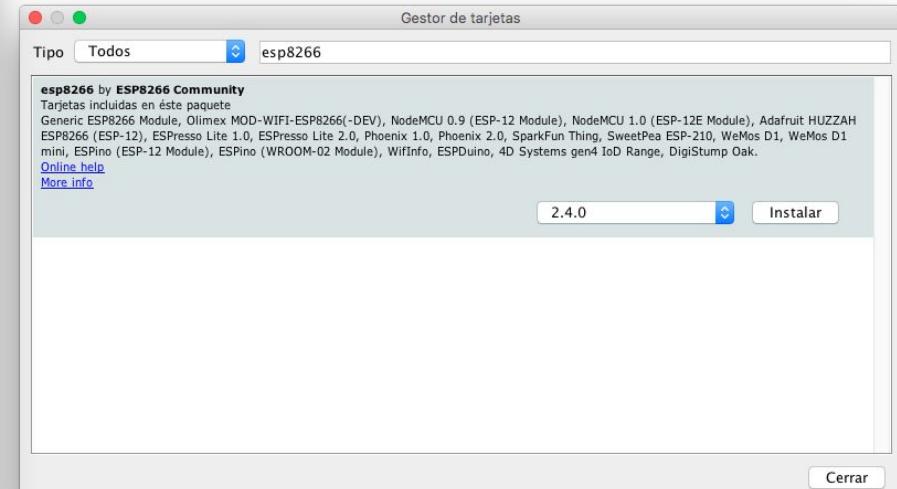
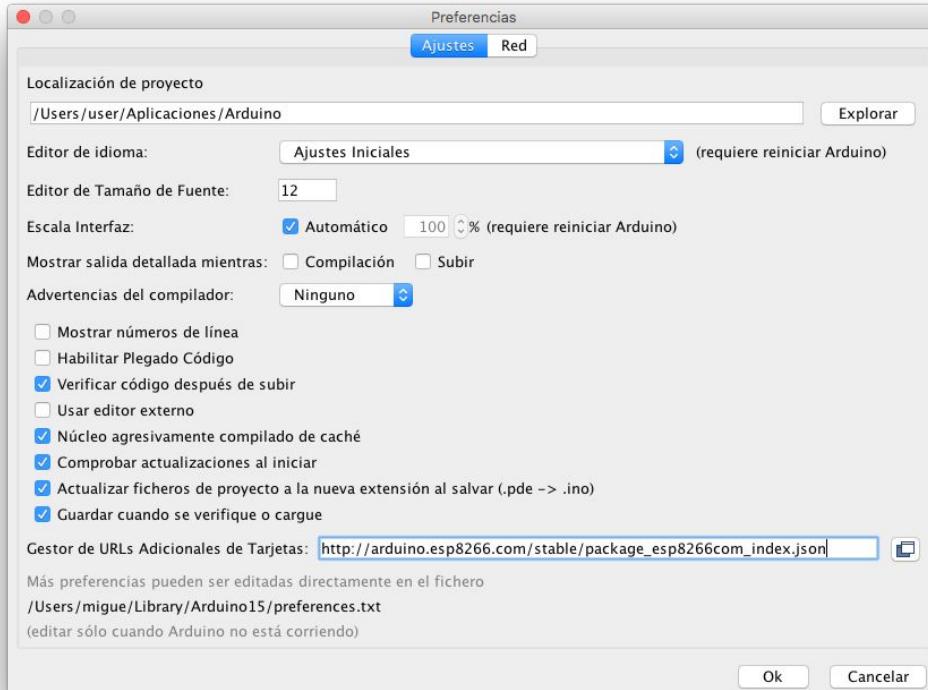


www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

2.2 Arduino



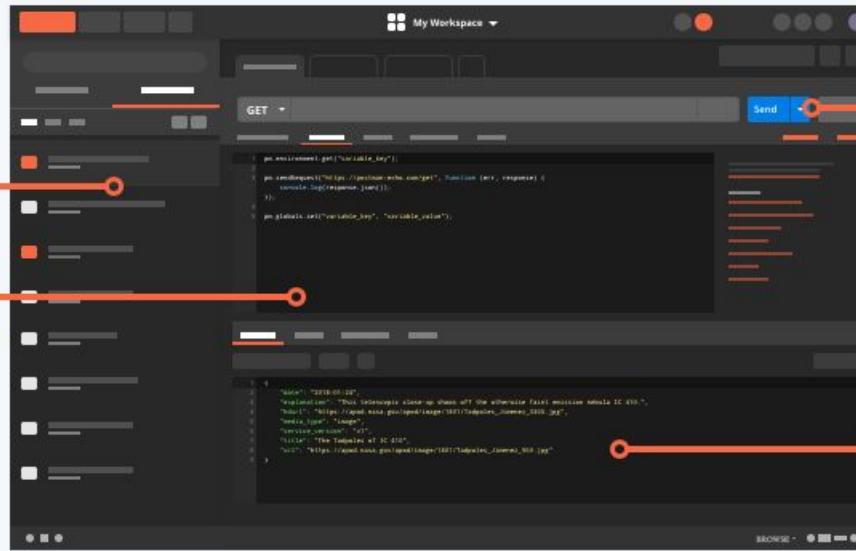
1



2

http://arduino.esp8266.com/stable/package_esp8266com_index.json

2.3 Postman



<https://www.getpostman.com>

2.3 Repositorio

This repository Search Pull requests Issues Marketplace Explore

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

FIWARE IoT integration using ESP866 board Edit Add topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Description	Time
FIWAREZone Uploaded arduino libraries		Latest commit 91b02ed a minute ago
Arduino Source	Uploaded arduino libraries	a minute ago
.DS_Store	Uploaded arduino libraries	a minute ago
README.md	Update README.md	17 hours ago
README.md		

IoT_Course

FIWARE IoT integration using ESP866 board

This course help developers to integrata a IoT development board by using NodeMCU with Arduino firmware. The communication protocol used by device is Ultralight2.0.

https://github.com/FIWAREZone/IoT_Course

03 >

Provisionamiento de dispositivo

3.1 Creando la entidad

HTTP POST:

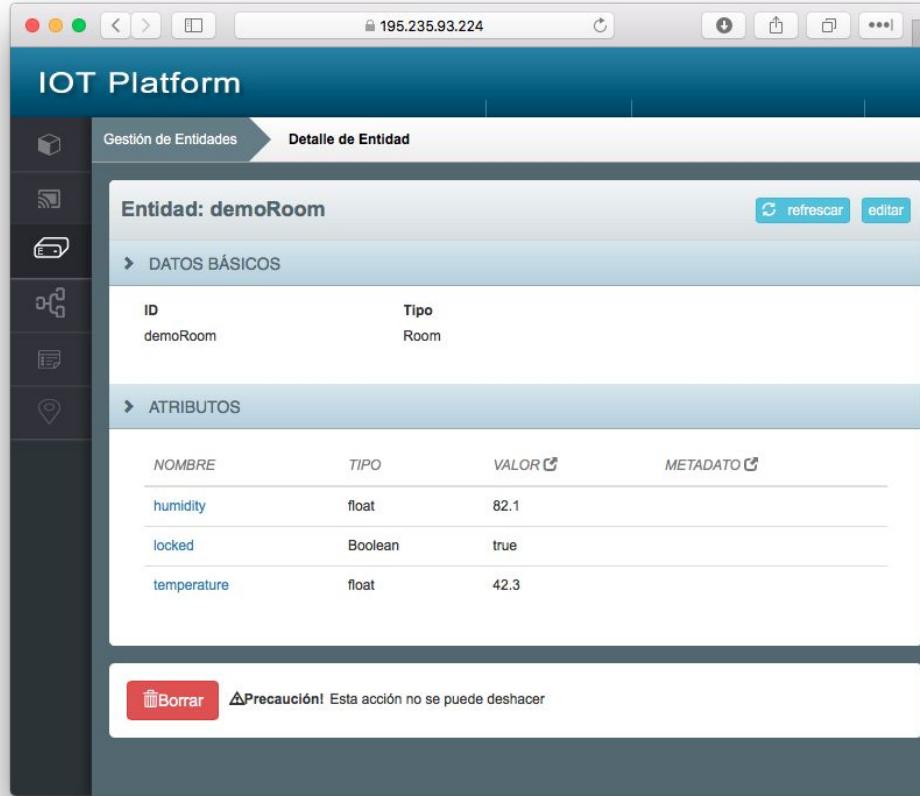
<https://{{host}}:10027/v2/entities>

Headers: {'content-type': 'application/json'; "Fiware-Service": {{service}}"; "Fiware-ServicePath":{{subservice}}"; "X-Auth-Token": {{token}}"}

Payload:

```
{  
    "id": "demoRoom",  
    "type": "Room",  
    "locked": {  
        "value": true,  
        "type": "Boolean"  
    },  
    "temperature": {  
        "value": 42.3,  
        "type": "float"  
    },  
    "humidity": {  
        "value": 82.1,  
        "type": "float"  
    }  
}
```

3.2 Creando la entidad



The screenshot shows the 'IOT Platform' interface with the URL '195.235.93.224' in the address bar. The main title is 'Detalle de Entidad' (Entity Detail) under 'Gestión de Entidades' (Entity Management). The entity being viewed is 'demoRoom'. The 'DATOS BÁSICOS' (Basic Data) section shows the ID 'demoRoom' and Type 'Room'. The 'ATRIBUTOS' (Attributes) section lists three attributes: 'humidity' (float, value 82.1), 'locked' (Boolean, value true), and 'temperature' (float, value 42.3). A red 'Borrar' (Delete) button is at the bottom left, with a warning message: 'Precaución! Esta acción no se puede deshacer' (Warning! This action cannot be undone).

NOMBRE	TIPO	VALOR	METADATO
humidity	float	82.1	
locked	Boolean	true	
temperature	float	42.3	

3.2 Creando el dispositivo

HTTP POST:

https://{{host_iota}}:8088/iot/devices

Headers: {'content-type': 'application/json'; "Fiware-Service": {{service}}"; "Fiware-ServicePath":{{subservice}}"; "X-Auth-Token": {{token}}'}

Payload:

```
{  
    "devices": [  
        {  
            "device_id": "id_sensor_demo",  
            "entity_name": "demoRoom",  
            "entity_type": "Room",  
            "attributes": [  
                { "object_id": "t", "name": "temperature", "type": "float" },  
                { "object_id": "h", "name": "humidity", "type": "float" }  
            ],  
            "lazy": [ ],  
            "commands": [  
                { "object_id": "conf", "name": "conf", "type": "command" }  
            ],  
            "static_attributes": [ ],  
            "protocol": "IoTA-UL",  
            "transport": "HTTP"  
        }  
    ]}
```

3.2 Creando el dispositivo

IOT Platform

Gestión de dispositivos > IoTAgent Ultralight 2.0 - Node.js > Detalle de un dispositivo

Dispositivo: id_sensor_demo

DATOS BÁSICOS

ID	Nombre De La Entidad	Tipo De Entidad
id_sensor_demo	demoRoom	Room
Entidad Relacionada		
demoRoom/Room		

MAPEO DE ATRIBUTOS

Valores por defecto

PARÁMETRO DE PROTOCOLO	ATRIBUTO DE DISPOSITIVO	TIPO

Valores establecidos para este dispositivo

PARÁMETRO DE PROTOCOLO	ATRIBUTO DE DISPOSITIVO	TIPO
t	temperature	float
h	humidity	float

3.2 Creando el dispositivo. Entidad

IOT Platform

Gestión de Entidades → Detalle de Entidad

Entidad: demoRoom

refrescar editar

DATOS BÁSICOS

ID	Tipo
demoRoom	Room

ATRIBUTOS

NOMBRE	TIPO	VALOR	METADATO
Timeinstant	ISO8601		
conf_info	commandResult		
conf_status	commandStatus	UNKNOWN	
humidity	float		
locked	Boolean	true	
temperature	float		
conf	command		<button>enviar comando</button>

3.3 Enviando datos por ultralight

HTTP POST:

http://{{host_iota}}:8085/iot/d?k={{UL_apikey}}&i={{device_ID}}

Headers: {'content-type': 'text/plain'}

Payload:

t|99#h|98

3.3 Enviando datos por ultralight

IOT Platform

Gestión de Entidades Detalle de Entidad

Entidad: demoRoom

refrescar editar

DATOS BÁSICOS

ID	Tipo
demoRoom	Room

ATRIBUTOS

NOMBRE	TIPO	VALOR	METADATO
TimeInstant	ISO8601	2018-04-05T11:26:03.00Z	
conf_info	commandResult		
conf_status	commandStatus	UNKNOWN	
humidity	float	98	TimeInstant: 2018-04-05T11:26:03.489Z
locked	Boolean	true	
status		FAIL	
temperature	float	99	TimeInstant: 2018-04-05T11:26:03.532Z
conf	command		

enviar comando

3.4 Enviando datos al dispositivo

HTTP POST:

https://{{host}}:10027/v1/contextEntities/type/{{device_type}}/id/{{entity_name}}/attributes/config

Headers: {'content-type': 'application/json'; "Fiware-Service": {{service}}"; "Fiware-ServicePath:{{subservice}}"; "X-Auth-Token: {{token}}"}

Payload:

```
{"value":{"paramName":"paramValue"},"type":"command"}
```

3.4 Enviando datos al dispositivo

IOT Platform

Gestión de Entidades Detalle de Entidad

Entidad: demoRoom

refrescar editar

DATOS BÁSICOS

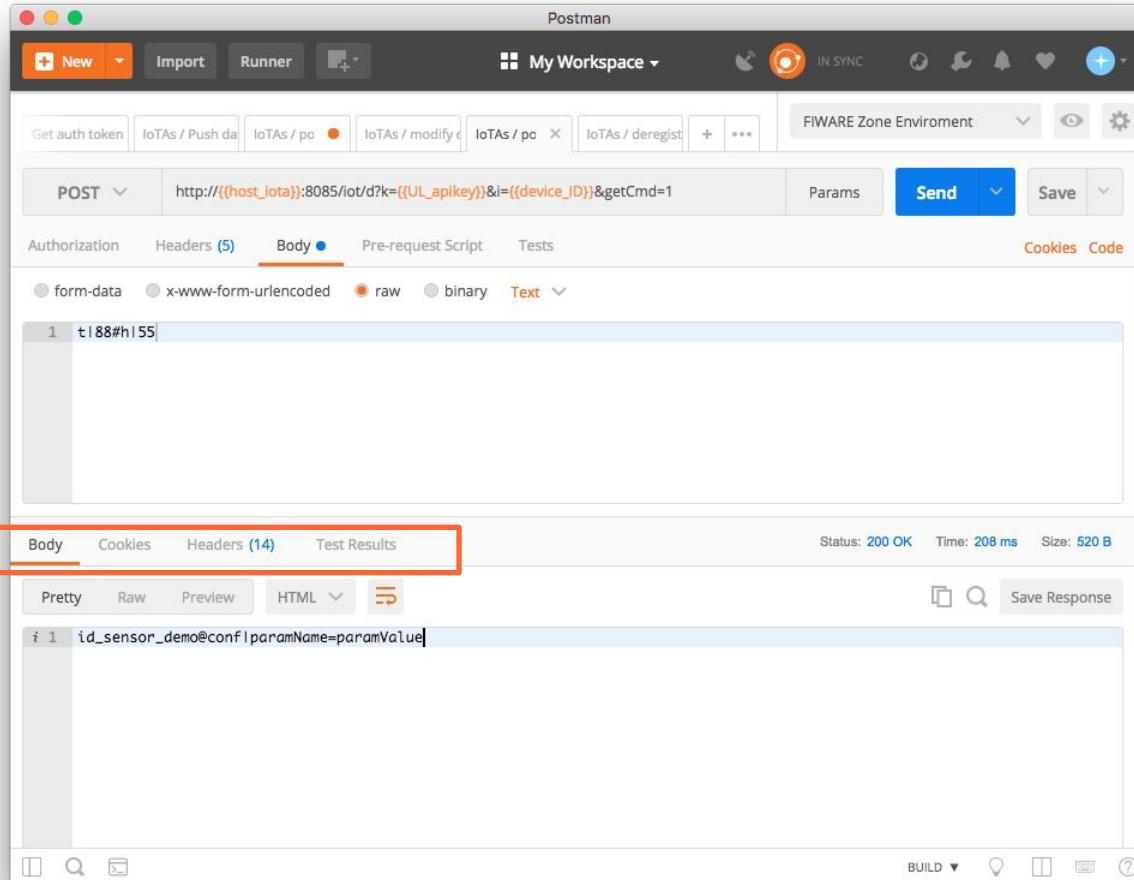
ID	Tipo
demoRoom	Room

ATRIBUTOS

NOMBRE	TIPO	VALOR	METADATO
TimeInstant	ISO8601	2018-04-05T11:27:39.00Z	
conf_info	commandResult		
conf_status	commandStatus	PENDING	TimeInstant: 2018-04-05T11:27:39.690Z
humidity	float	98	TimeInstant: 2018-04-05T11:26:03.489Z
locked	Boolean	true	
status		FAIL	
temperature	float	99	TimeInstant: 2018-04-05T11:26:03.532Z
conf	command		

enviar comando

3.4 Enviando datos al dispositivo



The screenshot shows the Postman application interface. At the top, there's a toolbar with 'New', 'Import', 'Runner', and 'My Workspace'. Below the toolbar, a header bar shows the URL: `http://{{host_iota}}:8085/iot/d?k={{UL_apikey}}&i={{device_ID}}&getCmd=1`. The main area has tabs for 'Body' (selected), 'Headers (5)', 'Pre-request Script', 'Tests', 'Cookies', and 'Code'. Under 'Body', the content type is set to 'Text' and the value is `t188#h155`. Below this, a red box highlights the 'Body', 'Cookies', 'Headers (14)', and 'Test Results' tabs. The 'Headers' tab shows 14 entries. The 'Test Results' tab shows a successful response with status `200 OK`, time `208 ms`, and size `520 B`. The response body is a JSON object with one item: `i 1 id_sensor_demo@config paramName=paramValue`.

3.4 Enviando datos al dispositivo

IOT Platform

Gestión de Entidades Detalle de Entidad

Entidad: demoRoom

DATOS BÁSICOS

ID	Tipo
demoRoom	Room

ATRIBUTOS

NOMBRE	TIPO	VALOR	METADATO
TimeInstant	ISO8601	2018-04-05T11:28:05.00Z	TimeInstant: 2018-04-05T11:28:05.178Z
conf_info	commandResult		TimeInstant: 2018-04-05T11:28:05.178Z
conf_status	commandStatus	DELIVERED	TimeInstant: 2018-04-05T11:28:05.178Z
humidity	float	55	TimeInstant: 2018-04-05T11:28:05.122Z
locked	Boolean	true	
status		FAIL	
temperature	float	88	TimeInstant: 2018-04-05T11:28:05.055Z
conf	command		<button>enviar comando</button>

04



Programa y funcionamiento

4.1 Variables de configuración

```
***** Global Variables *****
//FIWARE Variables
String FIWARE_device_ID = "SENSOR_ID";
String FIWARE_server = "195.235.93.235";
String FIWARE_port = "8085";
String FIWARE_apikey = "REEMPLAZAR_APIKEY";

//Wifi Variables
const char* WiFi_Network = "NOMBRE_WIFI";
const char* WiFi_Password = "PASS_WIFI";

const char* WiFi_SoftAP_Name = "FIWAREZone_IoT";
const char* WiFi_SoftAP_WiFi_Name = "FIWAREZone_IoT_Wifi";
```

4.2 Librerías empleadas

```
#include <FS.h>                                //this needs to be first, or it all crashes and burns

//ESP8266 Native libraries
#include <ESP8266WiFi.h>                         //https://github.com/esp8266/Arduino
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPUpdateServer.h>
#include <ESP8266HTTPClient.h>

//External libraries
#include <WiFiManager.h>                          //https://github.com/tzapu/WiFiManager
#include <ArduinoJson.h>                            //https://github.com/bblanchon/ArduinoJson
```

CONTENTS:

[Installing](#)[Reference](#)[Libraries](#)[File system](#)[ESP8266WiFi](#)[OTA Updates](#)[PROGMEM](#)[Boards](#)[FAQ](#)[Exception causes](#)[Debugging](#)[Stack Dump](#)[Using with Eclipse](#)

4.2.1 Librerías Nativas ESP8266

- [!\[\]\(4663682e3c2379a982181178e69b96ca_img.jpg\) ESP8266AVRISP](#)
- [!\[\]\(b985c7a2b649f83d0ae41b43712c07df_img.jpg\) ESP8266HTTPClient](#)
- [!\[\]\(b96f8226e35bcad959b0533b2efebc76_img.jpg\) ESP8266HTTPUpdateServer](#)
- [!\[\]\(b4fa684968a84cea43d4b420a952d2ab_img.jpg\) ESP8266LLMNR](#)
- [!\[\]\(69e984ef39ef510058fa1dfa8ef045cd_img.jpg\) ESP8266NetBIOS](#)
- [!\[\]\(32ec8fbdf47ec51a68812752f9bd6bcf_img.jpg\) ESP8266SSDP](#)
- [!\[\]\(01e410ef4e371379b82bb9d2e3877b8b_img.jpg\) ESP8266WebServer](#)
- [!\[\]\(a44e7b2bea71ef81080c4fa4bc6ca1ff_img.jpg\) ESP8266WiFi](#)
- [!\[\]\(cddfdc0808878faac8cdaa9c150ef721_img.jpg\) ESP8266WiFiMesh](#)
- [!\[\]\(bcd5aa7cf693d5eba8b1f98dbd4d35f3_img.jpg\) ESP8266httpUpdate](#)
- [!\[\]\(deb2d584faf5be12d7e59c343defaed8_img.jpg\) ESP8266mDNS](#)

<https://github.com/esp8266/Arduino>

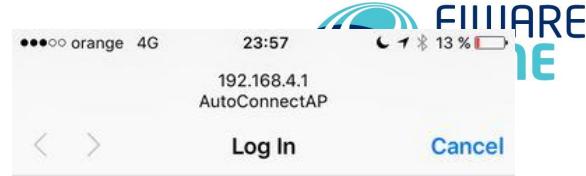
4.2.2 Librerías externas

WiFiManager

Permite configurar un punto de acceso. Si no lo encuentra, crea red propia para configurar el AP conectándose mediante wifi. Guarda las credenciales en la memoria interna

<https://github.com/tzapu/WiFiManager>

Telefónica



AutoConnectAP

WiFiManager

Configure WiFi

Configure WiFi (No Scan)

4.2.2 Librerías externas

ArduinoJson

Serialización y deserialización de JSON de forma sencilla para el envío de datos

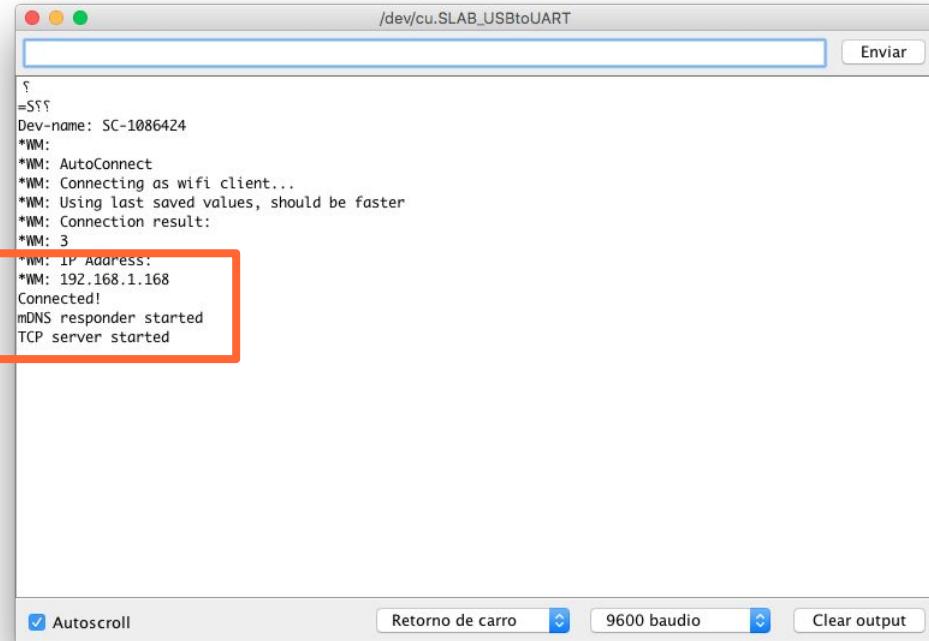


<https://github.com/bblanchon/ArduinoJson>

Telefónica

4.3 Configuración WiFi

Revisamos el monitor del puerto serie
 Ya está conectado a la Wifi que hemos
 configurado. Ahora podemos acceder
 desde el navegador, conectado a la misma
 red



```
/dev/cu.SLAB_USBtoUART
Enviar
?
=SSS
Dev-name: SC-1086424
*WM:
*WM: AutoConnect
*WM: Connecting as wifi client...
*WM: Using last saved values, should be faster
*WM: Connection result:
*WM: 3
*WM: IP Address:
*WM: 192.168.1.168
Connected!
mDNS responder started
TCP server started

Autoscroll Retorno de carro 9600 baudio Clear output
```

4.4 Servidor Web. Página principal

FIWARE.ino

```
void setup()
```

```
//Bind Webserver URL functions
server.on ( "/", handleRoot );
server.on ( "/wrst", handleWrst );
server.on ( "/wifi", handleWifi );
server.on ( "/webota", handleWebota );
server.on ( "/help", handleHelp );
server.on ( "/postul2", handlePostUL2 );
server.on ( "/postul2data", handlePostUL2data );
```

router.ino

Todas las funciones que renderizan las páginas

```
void handleRoot() {

    String page = FPSTR(HTTP_HEAD);
    page.replace("{v}", "FIWARE_Zone_IoT");
    page += FPSTR(HTTP_STYLE);
    page += FPSTR(HTTP_HEAD_END);
    page += F("<h1>FIWARE IoT Device</h1>");
    page += F("<h3>Menu</h3>");
    page += FPSTR(HTTP_MAIN_FORM);
    page += String("hostname:");
    page += String(dev_hostname);
    page += String(".local");
    page += FPSTR(HTTP_END);
    server.send(200, "text/html", page);

}
```

4.4 Servidor Web. Página principal

router.ino

```
const char HTTP_MAIN_FORM[ ]
```

```
const char HTTP_MAIN_FORM[] PROGMEM = "<form action=\"/wifi\" method=\"get\"><button>Setup Wifi</button></form><br/><form action=\"/wrst\" method=\"get\"><button>Clear wifi setup</button></form><br/>"
```

```
const char HTTP_MAIN_FORM[] PROGMEM =
<form action="/wifi" method="get"><button>Setup Wifi</button></form><br/>
<form action="/wrst" method="get"><button>Clear wifi setup</button></form><br/>
<form action="/webota" method="get"><button>Update</button></form><br/>
<form action="/postul2" method="get"><button>POST UL2</button></form><br/>
<form action="/help" method="get"><button>Help</button></form><br/>
```

Utilidades de interés:

Convertidor string C (caracteres de escape)

http://tomeko.net/online_tools/cpp_text_escape.php?lang=en

Formateador de HTML

<http://minifycode.com/html-beautifier/>

4.4 Servidor Web. Página principal



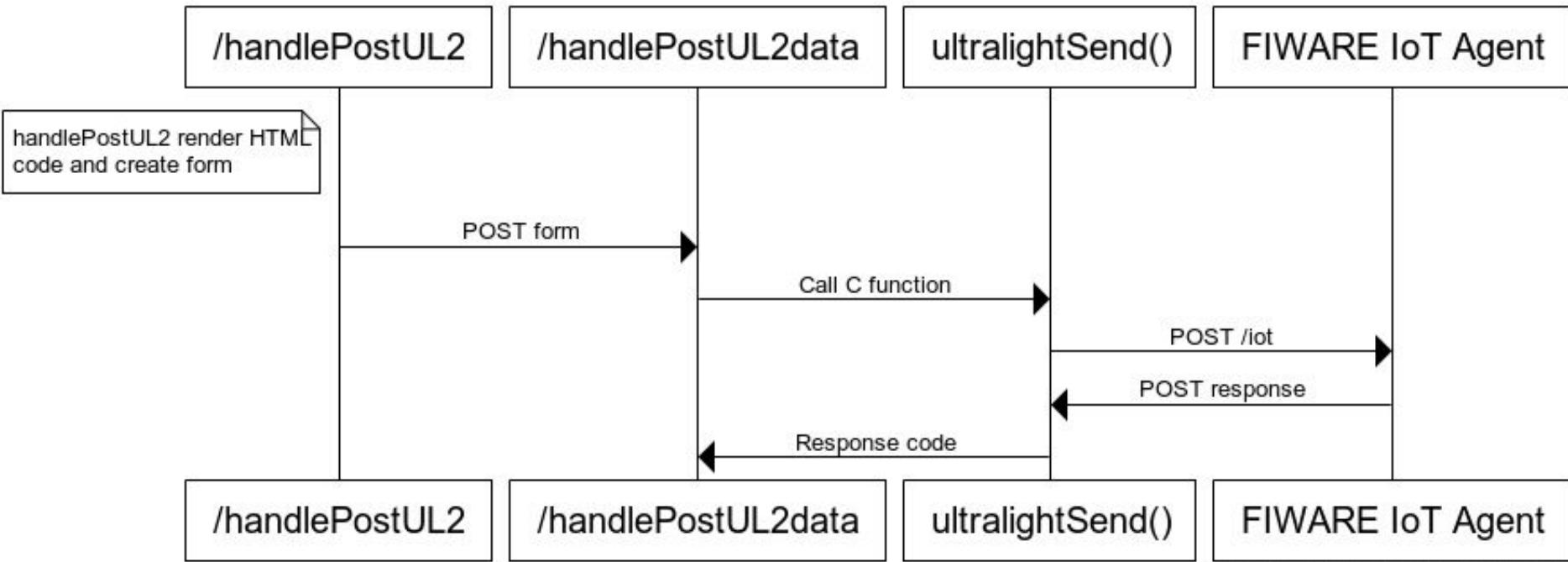
The screenshot shows a web browser window with the address bar displaying "192.168.1.168". The main content area is titled "FIWARE IoT Device" and contains a "Menu" section with five blue buttons:

- Setup Wifi
- Clear wifi setup
- Update
- POST ULL2
- Help

At the bottom of the page, the text "hostname:SC-1086424.local" is displayed.

4.5 Secuencia de envío

Data sequence



www.websequencediagrams.com

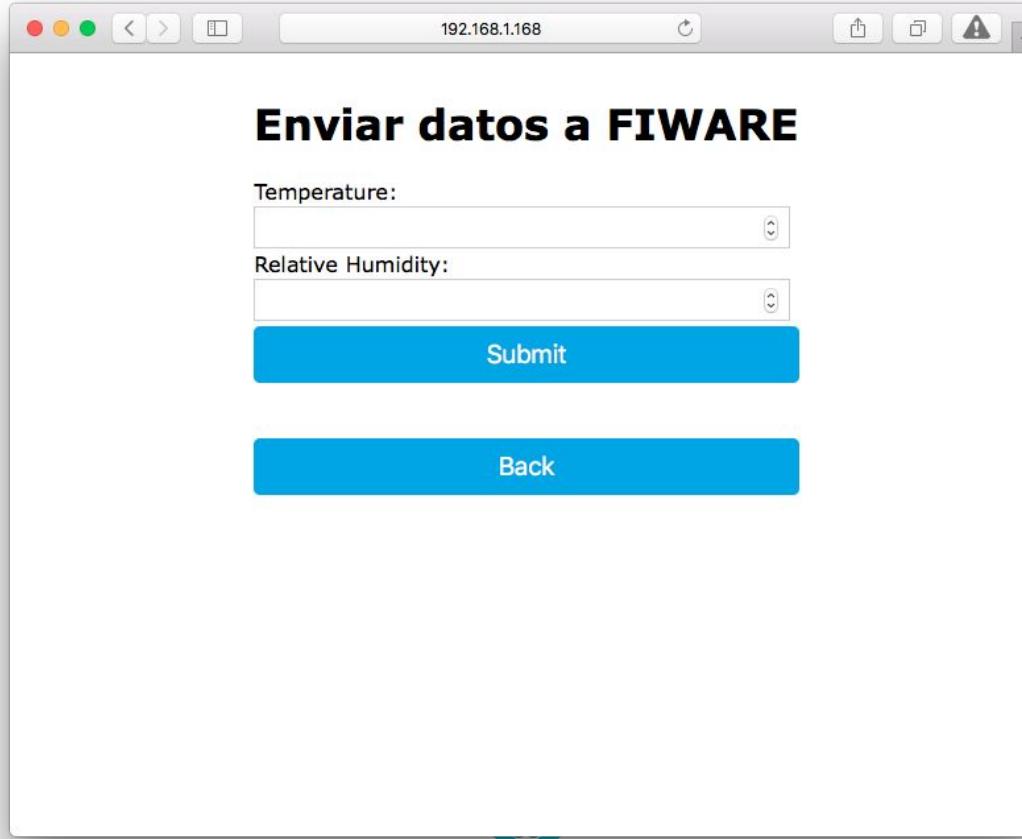
4.6.1 Función de envío a FIWARE

```
void handlePostUL2(){
    String page = FPSTR(HTTP_HEAD);
    page.replace("{v}", "Info");
    page += FPSTR(HTTP_SCRIPT);
    page += FPSTR(HTTP_STYLE);
    page += FPSTR(HTTP_HEAD_END);
    page += F("<h1>Enviar datos a FIWARE</h1>");
    page += F("<form action=\"/postul2data\" method=\"post\"> Temperature: <input type=\"number\"");
    page += FPSTR(HTTP_BACK_BUTTON);
    page += FPSTR(HTTP_END);
    server.send(200, "text/html", page);
}
```



```
<form action="/postul2data" method="post">Temperature:
    <input type="number" name="tmp" min="-273" max="999">
    <br>Relative Humidity:
    <input type="number" name="rh" min="0" max="100">
    <br>
    <button type="submit" value="Submit">Submit</button>
</form>
```

4.6.2 Panel de envío a FIWARE



Enviar datos a FIWARE

Temperature:

Relative Humidity:

Submit

Back

4.6.3 Llamada a función de envío

```
void handlePostUL2data(){  
  
    //Search form post values  
    String tmp, rh;  
    if (server.args() > 0 ) {  
        for ( uint8_t i = 0; i < server.args(); i++ ) {  
            if (server.argName(i) == "tmp") {  
                // do something here with value from server.arg(i);  
                tmp = server.arg(i);  
            }  
            else if(server.argName(i) == "rh"){  
                rh = server.arg(i);  
            }  
        }  
    }  
  
    //POST Data  
    String page = FPSTR(HTTP_HEAD);  
    page.replace("{v}", "Info");  
    page += FPSTR(HTTP_SCRIPT);  
    page += FPSTR(HTTP_STYLE);  
    page += FPSTR(HTTP_HEAD_END);  
    page += F("<h1>Enviando datos a FIWARE</h1>");  
  
    //Parse response  
    int returnCode = ultralightSend(FIWARE_server,FIWARE_port,FIWARE_token,FIWARE_ID,"temp|"+tmp+"#hr|"+rh);  
    if (returnCode ==200){  
        page += F("<h3>Resultado</h3>");  
        page += F("La peticion se ha ejecutado correctamente");  
    }  
    else{  
        page += F("<h3>Resultado</h3>");  
        page += F("Ha habido un error en la peticion");  
        page += String(returnCode);  
    }  
  
    //End page  
    page += FPSTR(HTTP_BACK_BUTTON);  
    page += FPSTR(HTTP_END);  
    server.send(200, "text/html", page);  
}
```

4.6.4 Función de envío por Ultralight 2.0

```
int ultralightSend (String URL, String port, String Token, String ID, String Body) {
    int httpCode = 0;

    Serial.println("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1");
    http.begin("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1"); //Specify request destination

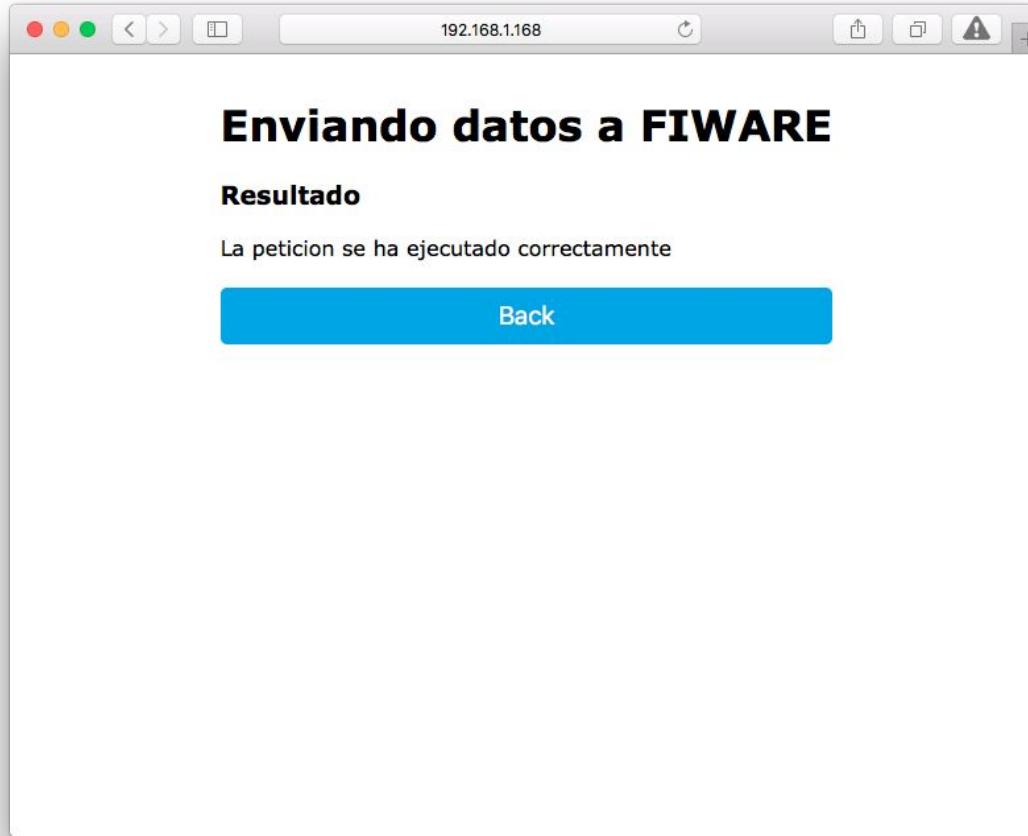
    http.addHeader("Content-Type", "text/plain"); //Specify content-type header

    httpCode = http.POST(Body); //Send the Body
    String payload = http.getString(); //Get the response payload

    Serial.println("Request Done");
    Serial.print("Return code: ");
    Serial.println(httpCode); //Print HTTP return code
    Serial.print("Response payload: ");
    Serial.println(payload); //Print request response payload

    http.end(); //Close connection
    return httpCode;
}
```

4.6.5 Respuesta de envío



The screenshot shows a web browser window with the address bar displaying "192.168.1.168". The main content area has a title "Enviando datos a FIWARE" and a section titled "Resultado" containing the message "La petición se ha ejecutado correctamente". A blue "Back" button is visible at the bottom of the page.

05



Otras funcionalidades

5.1 WiFi Manager

```
#define USE_CREDENTIALS Comentar
```

```
***** Global Variables *****/
//FIWARE Variables
String FIWARE_device_ID = "SENSOR_ID";
String FIWARE_server = "195.235.93.235";
String FIWARE_port = "8085";
String FIWARE_apikey = "REEMPLAZAR_APIKEY";

//Wifi Variables
const char* WiFi_Network = "NOMBRE_WIFI";
const char* WiFi_Password = "PASS_WIFI";

const char* WiFi_SoftAP_Name = 'FIWAREZone_IoT';
const char* WiFi_SoftAP_WiFi_Name = "FIWAREZone_IoT_Wifi";
```

Modificar

5.1 WiFi Manager



Primer inicio

La primera vez que arranca el dispositivo, al no tener una red configurada, este crea una red propia para configurar la red wifi a la que tiene que conectarse la placa. La red que crea tendrá el nombre de la variable definida como **WiFi_SoftAP_Name**.

La IP a la que hay que ir una vez conectados es :**192.168.4.1**



5.1 WiFi Manager

Configurar WiFi

Al entrar en Setup Wifi, el dispositivo se reiniciará en modo configuración de wifi creando una nueva red con el nombre de **WiFi_SoftAP_WiFi_Name**. Al entrar a esta nueva red, nos saldrá un portal cautivo en el que podemos configurar la red wifi.

En caso de que no salga dicho portal, nos conectamos a la misma IP de antes:

192.168.4.1

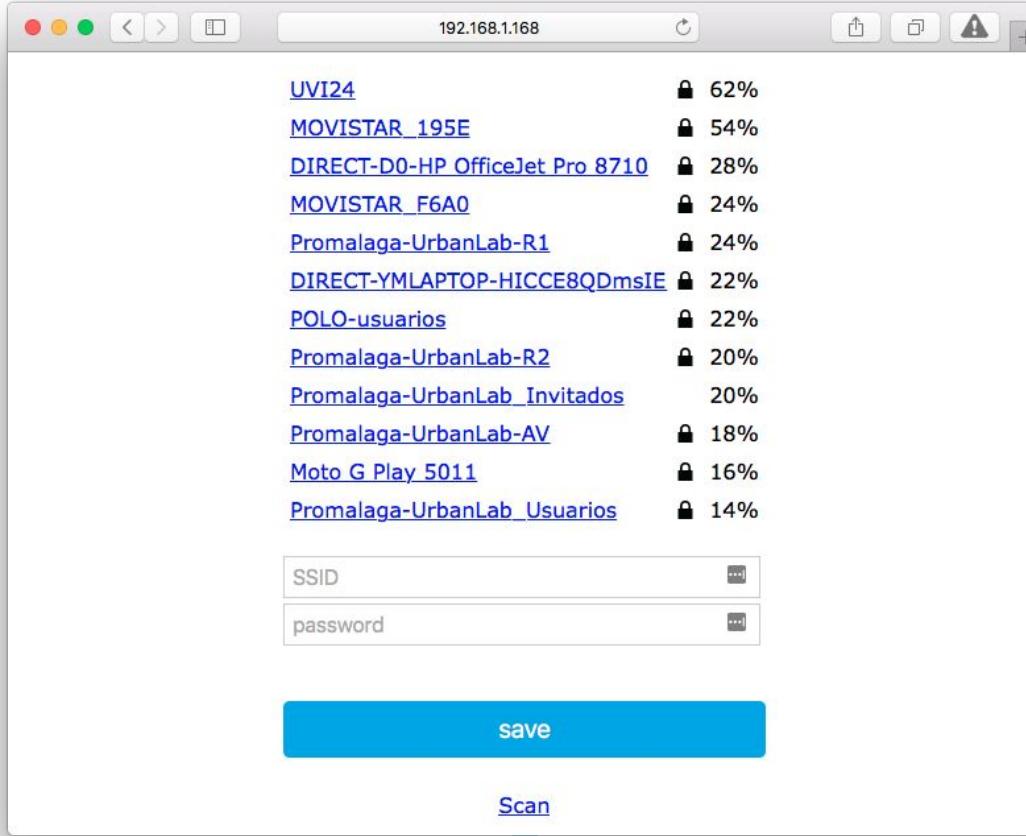


FIWARE-WiFi

WiFiManager

- [Configure WiFi](#)
- [Configure WiFi \(No Scan\)](#)
- [Info](#)
- [Exit](#)

5.1 WiFi Manager



The screenshot shows a web-based WiFi manager interface with the following details:

SSID	Status
UVI24	62%
MOVISTAR_195E	54%
DIRECT-D0-HP OfficeJet Pro 8710	28%
MOVISTAR_F6A0	24%
Promalaga-UrbanLab-R1	24%
DIRECT-YMLAPTOP-HICCE8QDmsIE	22%
POLO-usuarios	22%
Promalaga-UrbanLab-R2	20%
Promalaga-UrbanLab_Invitados	20%
Promalaga-UrbanLab-AV	18%
Moto G Play 5011	16%
Promalaga-UrbanLab_Usuarios	14%

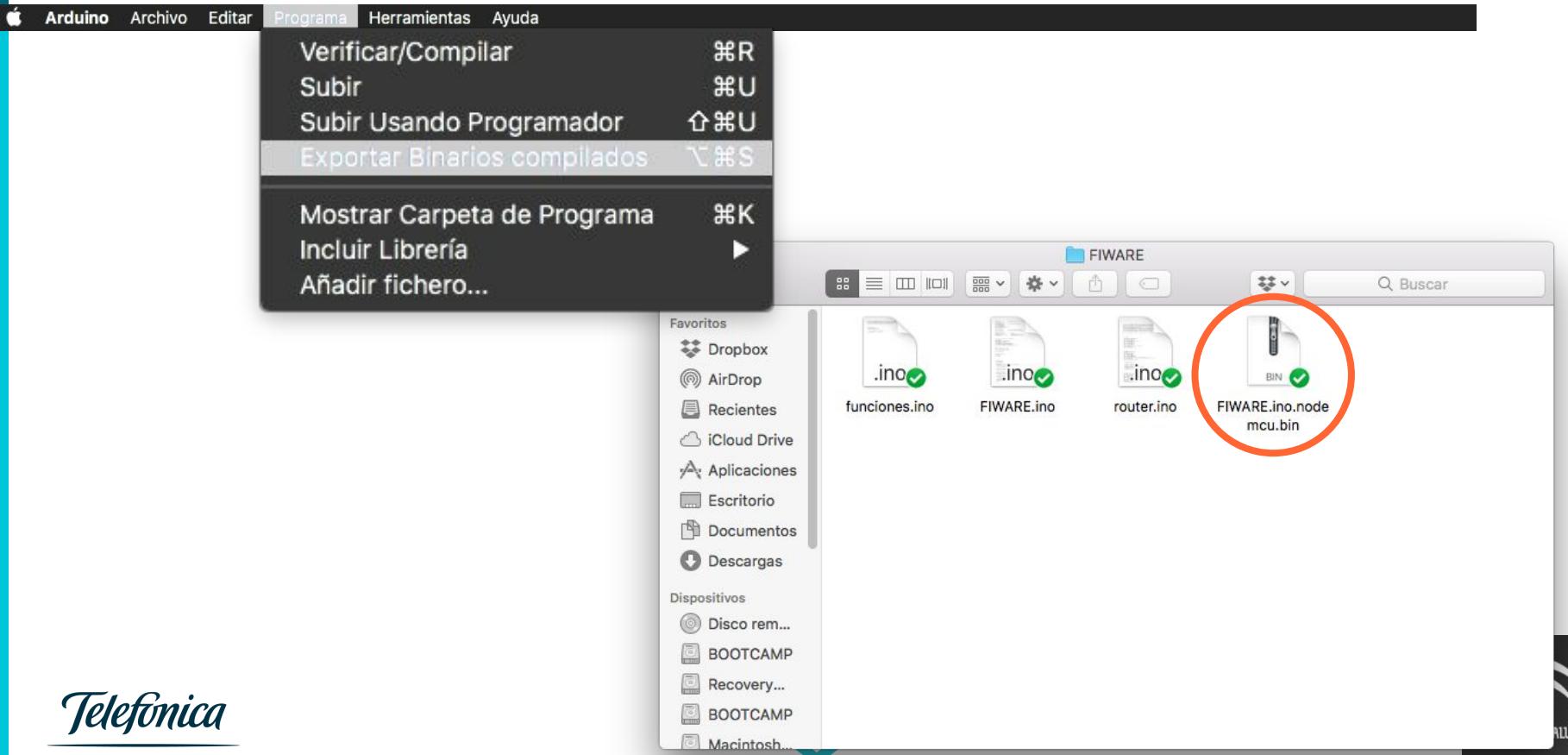
Below the list are two input fields: "SSID" and "password", each with a small icon to their right. At the bottom are two buttons: a large blue "save" button and a smaller "Scan" button.

5.2 OTA

```
//OTA
const char* update_path = "/webota";
const char* update_username = "admin";
const char* update_password = "admin";
```



5.2 OTA



5.3 Datalogger

Proyecto Datalogger.ino

Ejemplo que envía los datos periódicos de los datos de un sensor.

Lectura de un sensor por **I2C** modelo **BMP085** con las siguientes medidas:

- Presión
- Temperatura
- Altitud



5.3 Datalogger

```
// I2C Barometric sensor libraries
#include <Wire.h>
#include <Adafruit_BMP085.h>
#define SCL_PIN D1
#define SDA_PIN D2
Adafruit_BMP085 bmp;

unsigned Long previousMillis=0;
unsigned Long deltaMillis=15000; Tiempo de envío
void dataSenderCallback();
```

```
// I2C Sensor Initialization
Wire.pins(SDA_PIN, SCL_PIN);
Wire.begin(SDA_PIN, SCL_PIN);
if (!bmp.begin()) {
    Serial.println("No BMP180 / BMP085");// we dont wait for this
    while (1) {}
}
```

5.3 Datalogger

```
void loop() {
    //WebServer task
    server.handleClient();

    if (millis()>previousMillis+deltaMillis){
        previousMillis=millis();
        //Execute task
        dataSenderCallback();
    }
}

String dataReaderSensor(){
    String t = "T=" + String(bmp.readTemperature()) + " *C";
    String p = "P=" + String(bmp.readPressure()) + " Pa";
    String a = "A=" + String(bmp.readAltitude(101325)) + " m";// insert pressure at sea level

    Serial.println("Reading data from sensors:");
    Serial.println(t);
    Serial.println(p);
    Serial.println(a);

    String r = "t|" + String(bmp.readTemperature()) +"#p|" + String(bmp.readPressure()) +"#a|" + String(bmp.readAltitude(101325));
    return r;
}

void dataSenderCallback(){
    int returnCode = ultralightSend(FIWARE_server,FIWARE_port,FIWARE_apikey,FIWARE_device_ID,dataReaderSensor());
    if (returnCode ==200){
        Serial.println("Data sent successfully to FIWARE");
    }
    else{
        Serial.println("Failure sending data to FIWARE");
    }
}
```



FIWARE
ZONE

@FIWAREZone

fiware.zone@fiware.zone

Telefónica



JUNTA DE ANDALUCÍA

CONSEJERÍA DE EMPLEO, EMPRESA Y COMERCIO